1 - **Para usuario “reparacion”:**

CREATE USER ‘reparacion’@’localhost’ IDENTIFIED BY ‘password’;

GRANT ALL PRIVILEGES ON reparacion.\* TO ‘reparacion’@’localhost’;

**Para usuario “reparacion\_dn”:**

CREATE USER ‘reparacion\_dn’@’localhost’ IDENTIFIED BY ‘password’;

GRANT ALL PRIVILEGES ON reparacion\_dn.\* TO ‘reparacion\_dn’@’localhost’;

1.1 - **Para usuario 'reparacion\_select’ :**

CREATE USER ‘reparacion\_select’@’localhost’ IDENTIFIED BY ‘password’;

GRANT SELECT ON reparacion.\* TO ‘reparacion\_select’@’localhost’;

**Para usuario ‘reparacion\_dn\_select’:**

CREATE USER ‘reparacion\_dn\_select’@’localhost’ IDENTIFIED BY ‘password’;

GRANT SELECT ON reparacion\_dn.\* TO ‘reparacion\_dn\_select’@’localhost’;

**Para usuario ‘reparacion\_update’:**

CREATE USER ‘reparacion\_update’@’localhost’ IDENTIFIED BY ‘password’;

GRANT SELECT, UPDATE, DELETE ON reparacion.\* TO ‘reparacion\_update’@’localhost’;

**Para usuario ‘reparacion\_dn\_update’:**

CREATE USER ‘reparacion\_dn\_update’@’localhost’ IDENTIFIED BY ‘password’;

GRANT SELECT, UPDATE, DELETE ON reparacion\_dn.\* TO ‘reparacion\_dn\_update’@’localhost’;

**Para usuario 'reparacion\_schema’ :**

CREATE USER 'reparacion\_schema’@’localhost’ IDENTIFIED BY ‘password’;

GRANT SELECT, UPDATE, DELETE,CREATE,DROP,ALTER,CREATE VIEW,SHOW VIEW,INDEX,CREATE ROUTINE,ALTER ROUTINE,TRIGGER ON reparacion.\* TO 'reparacion\_schema’@’localhost’;

**Para usuario 'reparacion\_dn\_schema’:**

CREATE USER 'reparacion\_dn\_schema’@’localhost’ IDENTIFIED BY ‘password’;

GRANT SELECT, UPDATE, DELETE,CREATE,DROP,ALTER,CREATE VIEW,SHOW VIEW,INDEX,CREATE ROUTINE,ALTER ROUTINE,TRIGGER ON reparacion\_dn.\* TO 'reparacion\_dn\_schema’@’localhost’;

2 - Listar dni, nombre y apellido de todos los clientes ordenados por dni en forma ascendente

* En ‘reparacion’: SELECT dniCliente,nombreApellidoCliente FROM cliente ORDER BY dniCliente ASC; (20000 rows in set (0.23 sec))
* En ‘reparacion\_dn’: SELECT DISTINCT(dniCliente),nombreApellidoCliente FROM reparacion ORDER BY dniCliente ASC; (20000 rows in set (5.75 sec))

La tabla reparacion\_dn puede tener clientes repetidos, por lo tanto, la cantidad de tuplas obtenidas puede ser mayor a la cantidad de clientes que realmente existen. La performance es más baja en reparacion\_dn.

3 - Hallar aquellos clientes que para todas sus reparaciones siempre hayan usado su tarjeta de crédito primaria (nunca la tarjeta secundaria).

* En ‘reparacion’:

SELECT dniCliente, nombreApellidoCliente

FROM cliente c  
 WHERE NOT EXISTS ( SELECT \*

FROM reparacion r

WHERE c.dniCliente = r.dniCliente AND  
 r.tarjetaReparacion = c.tarjetaPrimaria);

* En ‘reparacion\_dn’:

SELECT dniCliente, nombreApellidoCliente

FROM reparacion r  
WHERE NOT EXISTS ( SELECT \*

FROM reparacion re

WHERE r.dniCliente = re.dniCliente AND

r.tarjetaReparacion = re.tarjetaPrimaria);

GROUP BY dniCliente;

4 -

**-Vista para la base de datos ‘reparacion’:**CREATE VIEW sucursalesPorCliente AS

SELECT c.dniCliente, s.codSucursal

FROM cliente c INNER JOIN sucursal s ON (c.ciudadCliente=s.ciudadSucursal)

**-Vista para la base de datos ‘reparacion\_dn’:**

CREATE VIEW clientes AS

SELECT DISTINCT dniCliente, ciudadCliente

FROM reparacion;

CREATE VIEW sucursales AS

SELECT DISTINCT codSucursal, ciudadSucursal

FROM reparacion;

CREATE VIEW sucursalesPorCliente AS  
SELECT dniCliente, codSucursal  
FROM clientes INNER JOIN sucursales ON (ciudadCliente=ciudadSucursal)

WHERE ciudadCliente = ciudadSucursal;

5 -

a-

SELECT dniCliente, nombreApellidoCliente  
FROM cliente c

WHERE (SELECT count(codSucursal)  
 FROM sucursal s

WHERE c.ciudadCliente=s.ciudadSucursal) = (SELECT count(DISTINCT su.codSucursal)

FROM reparacion r INNER JOIN   
 sucursal su ON (r.codSucursal=su.codSucursal)  
WHERE c.ciudadCliente = su.ciudadSucursal AND c.dniCliente=r.dniCliente)

LIMIT 100

b-SELECT dniCliente, nombreApellidoCliente  
FROM cliente c

WHERE (SELECT count(codSucursal)  
 FROM sucursalesPorCliente

WHERE c.dniCliente=sucursalesPorCliente.dniCliente) = (SELECT count(DISTINCT su.codSucursal)

FROM reparacion r INNER JOIN   
 sucursal su ON (r.codSucursal=su.codSucursal)  
WHERE c.ciudadCliente = su.ciudadSucursal AND c.dniCliente=r.dniCliente)

LIMIT 100;

6 - Hallar los clientes que en alguna de sus reparaciones hayan dejado como dato de contacto el mismo domicilio y ciudad que figura en su DNI.

**-Consulta en la bd ‘reparacion’:**

SELECT DISTINCT(c.dniCliente), c.nombreApellidoCliente

FROM reparacion r INNER JOIN cliente c ON (r.dniCliente=c.dniCliente)  
WHERE r.direccionReparacionCliente=c.domicilioCliente AND  
 r.ciudadReparacionCliente=c.ciudadCliente;

SELECT DISTINCT(c.dniCliente), c.nombreApellidoCliente

FROM reparacion r INNER JOIN cliente c ON (r.dniCliente=c.dniCliente AND r.direccionReparacionCliente=c.domicilioCliente AND r.ciudadReparacionCliente=c.ciudadCliente);

**-Consulta en la bd ‘reparacion\_dn’:**

SELECT DISTINCT(dniCliente), nombreApellidoCliente

FROM reparacion

WHERE direccionReparacionCliente=domicilioCliente AND ciudadReparacionCliente=ciudadCliente;

7 -Para aquellas reparaciones que tengan registrados mas de 3 repuestos, listar el DNI del cliente, el código de sucursal, la fecha de reparación y la cantidad de repuestos utilizados.

**-Consulta en la bd ‘reparacion’:**

SELECT r.dniCliente, r.codSucursal, r.fechaInicioReparacion, count(rr.repuestoReparacion)

FROM reparacion r INNER JOIN repuestoreparacion rr ON (r.dniCliente=rr.dniCliente AND r.fechaInicioReparacion=rr.fechaInicioReparacion)

GROUP BY r.dniCliente, r.fechaInicioReparacion  
HAVING count(rr.repuestoReparacion) > 3;

**-Consulta en la bd ‘reparacion\_dn’:**

SELECT r.dniCliente, r.codSucursal, r.fechaInicioReparacion, count(DISTINCT(r.repuestoReparacion))

FROM reparacion r

GROUP BY r.dniCliente, r.fechaInicioReparacion  
HAVING count(DISTINCT (r.repuestoReparacion)) > 3;

8 - CREATE TABLE `reparacionesporcliente` (

`id` INT(11) UNSIGNED NOT NULL AUTO\_INCREMENT,

`dniCliente` INT(11) NOT NULL,

`cantidadReparaciones` INT(11) NOT NULL,

`fechaUltimaActualizacion` DATETIME NOT NULL,

`usuario` CHAR(16) NOT NULL,

PRIMARY KEY (`id`)

);

9 -

a -

USE reparacion;

DELIMITER //

CREATE PROCEDURE puntoNueve()

BEGIN

DECLARE fin BOOLEAN DEFAULT FALSE;

DECLARE cantReparaciones INT;

DECLARE dniC INT;

DECLARE reparacionesCliente CURSOR FOR SELECT count(\*) as cantidad, dniCliente FROM reparacion group by dniCliente;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET fin = TRUE;

START TRANSACTION;

OPEN reparacionesCliente;

ciclo\_loop: LOOP

FETCH reparacionesCliente INTO cantReparaciones, dniC;

IF fin THEN

LEAVE ciclo\_loop;

END IF;

INSERT INTO reparacionesporcliente (dniCliente,cantidadReparaciones,fechaUltimaActualizacion,usuario) VALUES (dniC,cantReparaciones,NOW(),USER());

END LOOP;

CLOSE reparacionesCliente;

COMMIT;

END //

b -

CALL puntoNueve();